Ref:

https://stackoverflow.com/questions/3476613/updating-one-field-in-jpa-entity

One to many and many to one Bidirectional Stud and University

@Override

@Transactional

public void salaryDetails(Integer eid, Salary salary) {

//Session session = sessionFactory.getCurrentSession();

org.hibernate.Session session = (Session)entityManager.getDelegate();

List<Salary> list = employeeSalaryData(eid);

for(Salary sal:list) {

sal.setSal(sal.getSal());

session.update(sal);

}

}

public List<Salary> employeeSalaryData(Integer eid) {

//Session session = sessionFactory.getCurrentSession();

org.hibernate.Session session = (Session)entityManager.getDelegate();

String queryStr = "select sal from Salary where empid = "+eid;

System.out.println("print query::::::::::"+queryStr.toString());

Query<Salary> query = session.createQuery(queryStr);

List<Salary> list = new ArrayList();

list = query.list();

return list;

}

package com.stud.model;

import java.io.Serializable;

import java.util.HashSet;

import java.util.Set;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.ManyToOne;

import javax.persistence.Table;

import org.springframework.cache.annotation.Cacheable;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name ="stud")

@JsonIgnoreProperties({"hibernateLazyInitializer","handler"})

public class Stud implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@GeneratedValue(strategy=GenerationType.AUTO)

@Column(name = "student\_id")

private long student\_id;

@Column(name = "first\_name")

private String first\_name;

@Column(name = "last\_name")

private String last\_name;

@Column(name = "section")

private String section;

@Column(name = "university\_id")

private long university\_id;

@JsonIgnore

@ManyToOne(cascade=CascadeType.ALL)

@JoinColumn(name="university\_id", nullable = false,insertable=false,updatable=false)

private University university;

public Stud() {

}

public Stud(String first\_name, String last\_name, String section) {

super();

this.first\_name = first\_name;

this.last\_name = last\_name;

this.section = section;

}

public long getStudent\_id() {

return student\_id;

}

public void setStudent\_id(long student\_id) {

this.student\_id = student\_id;

}

public String getFirst\_name() {

return first\_name;

}

public void setFirst\_name(String first\_name) {

this.first\_name = first\_name;

}

public String getLast\_name() {

return last\_name;

}

public void setLast\_name(String last\_name) {

this.last\_name = last\_name;

}

public String getSection() {

return section;

}

public void setSection(String section) {

this.section = section;

}

public University getUniversity() {

return university;

}

public void setUniversity(University university) {

this.university = university;

}

public long getUniversity\_id() {

return university\_id;

}

public void setUniversity\_id(long university\_id) {

this.university\_id = university\_id;

}

@Override

public String toString() {

return "Stud [student\_id=" + student\_id + ", first\_name=" + first\_name + ", last\_name=" + last\_name

+ ", section=" + section + ", university=" + university + "]";

}

}

package com.stud.model;

import java.io.Serializable;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.FetchType;

import javax.persistence.GeneratedValue;

import javax.persistence.GenerationType;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.OneToMany;

import javax.persistence.Table;

import javax.xml.bind.annotation.XmlAccessType;

import javax.xml.bind.annotation.XmlAccessorType;

import javax.xml.bind.annotation.XmlRootElement;

import com.fasterxml.jackson.annotation.JsonAutoDetect;

import com.fasterxml.jackson.annotation.JsonIgnore;

import com.fasterxml.jackson.annotation.JsonIgnoreProperties;

@Entity

@Table(name = "university")

@JsonIgnoreProperties({"hibernateLazyInitializer","handler"})

@JsonAutoDetect

public class University implements Serializable{

private static final long serialVersionUID = 1L;

@Id

// @GeneratedValue(strategy=GenerationType.AUTO)

@Column(name = "university\_id")

private long university\_id;

@Column(name = "name")

private String name;

@Column(name = "country")

private String country;

// @OneToMany(cascade=CascadeType.ALL,mappedBy="Stud")

//@OneToMany(cascade=CascadeType.ALL,mappedBy="university")

@OneToMany(fetch = FetchType.EAGER,targetEntity = Stud.class,mappedBy="university")

private List<Stud> stud;

public University() {

}

public List<Stud> getStud() {

return stud;

}

public void setStud(List<Stud> stud) {

this.stud = stud;

}

public University(String name, String country) {

this.name = name;

this.country = country;

}

public long getId() {

return university\_id;

}

public void setId(long id) {

this.university\_id = id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getCountry() {

return country;

}

public void setCountry(String country) {

this.country = country;

}

@Override

public String toString() {

return "University [id=" + university\_id + ", name=" + name + ", country=" + country + "]";

}

}

Many to Many (Student subjects)

package com.stud.model;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.GeneratedValue;

import javax.persistence.Id;

import javax.persistence.JoinColumn;

import javax.persistence.JoinTable;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name = "studentmany")

public class StudManytoMany implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@Column(name = "studentid")

private long studentid;

@Column(name = "firstname")

private String firstname;

@Column(name = "lastname")

private String lastname;

@ManyToMany(cascade = CascadeType.ALL)

@JoinTable(name = "studentsubject",

joinColumns = { @JoinColumn(name = "studentid", referencedColumnName = "studentid") },

inverseJoinColumns = { @JoinColumn(name = "subjectid" ,referencedColumnName = "subjectid") })

private List<Subject> subjects = new ArrayList<Subject>();

public StudManytoMany() {

}

public StudManytoMany(long studentid,String firstname, String lastname) {

this.studentid=studentid;

this.firstname = firstname;

this.lastname = lastname;

}

public long getStudentid() {

return studentid;

}

public void setStudentid(long studentid) {

this.studentid = studentid;

}

public String getFirstname() {

return firstname;

}

public void setFirstname(String firstname) {

this.firstname = firstname;

}

public String getLastname() {

return lastname;

}

public void setLastname(String lastname) {

this.lastname = lastname;

}

public List<Subject> getSubjects() {

return subjects;

}

public void setSubjects(List<Subject> subjects) {

this.subjects = subjects;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + (int) (studentid ^ (studentid >>> 32));

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (!(obj instanceof StudManytoMany))

return false;

StudManytoMany other = (StudManytoMany) obj;

if (studentid != other.studentid)

return false;

return true;

}

@Override

public String toString() {

return "StudManytoMany [id=" + studentid + ", firstName=" + firstname + ", lastName="

+ lastname + "]";

}

}

Subjects:

package com.stud.model;

import java.io.Serializable;

import java.util.ArrayList;

import java.util.List;

import javax.persistence.CascadeType;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.ManyToMany;

import javax.persistence.Table;

@Entity

@Table(name = "subject")

public class Subject implements Serializable{

private static final long serialVersionUID = 1L;

@Id

@Column(name = "subjectid")

private long subjectid;

@Column(name = "name")

private String name;

@ManyToMany(mappedBy="subjects",cascade = CascadeType.ALL)

private List<StudManytoMany> students = new ArrayList<StudManytoMany>();

public Subject(){

}

public Subject(long subjectid,String name){

this.subjectid=subjectid;

this.name = name;

}

public long getSubjectid() {

return subjectid;

}

public void setSubjectid(long subjectid) {

this.subjectid = subjectid;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public List<StudManytoMany> getStudents() {

return students;

}

public void setStudents(List<StudManytoMany> students) {

this.students = students;

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + (int) (subjectid ^ (subjectid >>> 32));

result = prime \* result + ((name == null) ? 0 : name.hashCode());

return result;

}

@Override

public boolean equals(Object obj) {

if (this == obj)

return true;

if (obj == null)

return false;

if (!(obj instanceof Subject))

return false;

Subject other = (Subject) obj;

if (subjectid != other.subjectid)

return false;

if (name == null) {

if (other.name != null)

return false;

} else if (!name.equals(other.name))

return false;

return true;

}

@Override

public String toString() {

return "Subject [id=" + subjectid + ", name=" + name + "]";

}

}

Ehcache with xml file

Refer:

<https://www.concretepage.com/spring-4/spring-4-ehcache-configuration-example-with-cacheable-annotation>

**package** com.stud.config;

**import** org.springframework.cache.CacheManager;

**import** org.springframework.cache.annotation.EnableCaching;

**import** org.springframework.cache.ehcache.EhCacheCacheManager;

**import** org.springframework.cache.ehcache.EhCacheManagerFactoryBean;

**import** org.springframework.context.annotation.Bean;

**import** org.springframework.context.annotation.Configuration;

**import** org.springframework.core.io.ClassPathResource;

@EnableCaching

@Configuration

**public** **class** EhCacheConfiguration {

@Bean

**public** CacheManager getEhCacheManager(){

**return** **new** EhCacheCacheManager(getEhCacheFactory().getObject());

}

@Bean

**public** EhCacheManagerFactoryBean getEhCacheFactory(){

EhCacheManagerFactoryBean factoryBean = **new** EhCacheManagerFactoryBean();

factoryBean.setConfigLocation(**new** ClassPathResource("ehcache.xml"));

factoryBean.setShared(**true**);

**return** factoryBean;

}}

Dependencies

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context-support</artifactId>

<version>4.2.8.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.7.RELEASE</version>

</dependency>

<dependency>

<groupId>net.sf.ehcache</groupId>

<artifactId>ehcache</artifactId>

<version>2.10.5</version>

</dependency>

http://websystique.com/spring/spring-4-cacheable-cacheput-cacheevict-caching-cacheconfig-enablecaching-tutorial/

**In ServiceImpl Add**

@Cacheable(value = "studcache")

@Transactional

@Override

**public** Stud getStudentById(**long** id) {

Stud student = studentDao.getStudentById(id);

**return** student;

}

Note: timeToIdleSeconds = 3. Then the object will be invalidated if it hasn't been requested for 4 seconds.

If timeToLiveSeconds = 90, then the object will be removed from cache after 90 seconds, even if it has been requested few milliseconds in the 90th second of its short life.

**Faced Problem in Ehcache**

Error: CandidateComponentsIndexLoader not found bean creation failed

Solution

1. Open the project's properties (e.g., right-click on the project's name in the project explorer and select "Properties").
2. Select "Deployment Assembly".
3. Click the "Add..." button on the right margin.
4. Select "Java Build Path Entries" from the menu of Directive Type and click "Next".
5. Select "Maven Dependencies" from the Java Build Path Entries menu and click "Finish".

<https://stackoverflow.com/questions/6210757/java-lang-classnotfoundexception-org-springframework-web-context-contextloaderl>

**Ehcahe without xml file :**

**Refer:** [**https://stackoverflow.com/questions/21944202/using-ehcache-in-spring-4-without-xml**](https://stackoverflow.com/questions/21944202/using-ehcache-in-spring-4-without-xml)

**http://www.ehcache.org/documentation/2.8/configuration/configuration.html**

[**http://www.ehcache.org/documentation/2.8/configuration/fast-restart**](http://www.ehcache.org/documentation/2.8/configuration/fast-restart)

**EhCacheConfiguration**

/\* CacheConfiguration cacheConfiguration = new CacheConfiguration();

cacheConfiguration.setName("studcache");

cacheConfiguration.setMemoryStoreEvictionPolicy("LRU");

cacheConfiguration.setMaxEntriesLocalHeap(1000);\*/

@Bean(destroyMethod = "shutdown")

public net.sf.ehcache.CacheManager cacheManager() {

CacheConfiguration employeeCacheConfig =

new CacheConfiguration("employee123", 1000)

.timeToLiveSeconds(3600)

.timeToIdleSeconds(600)

.eternal(false)

.persistence(new PersistenceConfiguration().strategy(Strategy.LOCALTEMPSWAP))

.memoryStoreEvictionPolicy(MemoryStoreEvictionPolicy.FIFO);

CacheConfiguration salaryCacheConfig =

new CacheConfiguration("salary123", 1000)

.timeToLiveSeconds(50)

.timeToIdleSeconds(100)

.eternal(false)

.persistence(new PersistenceConfiguration().strategy(Strategy.LOCALTEMPSWAP))

.memoryStoreEvictionPolicy(MemoryStoreEvictionPolicy.LFU);

**DiskStoreConfiguration dsc = new DiskStoreConfiguration();**

**dsc.setPath("D:\\data\\ehcache");**

net.sf.ehcache.config.Configuration config = new net.sf.ehcache.config.Configuration();

config.addCache(employeeCacheConfig);

config.addCache(salaryCacheConfig);

**config.addDiskStore(dsc);**

return net.sf.ehcache.CacheManager.newInstance(config);

}

Single colum update

@DynamicUpdate annotation to be added in model class

@Override

**public** **void** UpdateSection(**long** student\_id) {

org.hibernate.Session session = (Session)entityManager.getDelegate();

Stud s1 = getStudentById(student\_id);

s1.setSection("Hibernate");

session.update(s1);

}

Ref: <https://dzone.com/articles/hibernate-query-example-qbe>

<http://websystique.com/spring/spring-4-cacheable-cacheput-cacheevict-caching-cacheconfig-enablecaching-tutorial/>

Difference bw spring cache n hibernate cache

 @Transient fields will be cached by Spring but not by Hibernate

,